Uma abordagem para o desenvolvimento de jogos adaptáveis a diversos controladores

Rodrigo Souza
Indigente - Universidade Federal da Bahia

Humberto Bandeira
Indigente - Universidade Federal da Bahia

Resumo

Este trabalho apresenta uma abordagem de desenvolvimento de *software* que facilita o uso de diversos dispositivos de entrada em jogos. O desenvolvedor programa o jogo para um dispositivo hipotético, definido de acordo com as ações do jogo e, a partir daí, mapeia eventos de teclado, mouse etc. em eventos do dispositivo hipotético. Mapeamentos recorrentes podem ser reusados. Com isso é mais fácil adaptar jogos a diversas possibilidades de interação, como gestos de mouse e reconhecimento de voz, e a dispositivos avançados, como luvas de realidade virtual e sistemas de rastreamento.

Keywords:: Interface humano-computador, realidade virtual, arquitetura de *software*

Author's Contact:

{rodrigo,nkbeto}@dcc.ufba.br

1 Introdução

Existem vários motivos para um jogador querer mudar a maneira como interage com um jogo [Myers et al. 2000]. Usar um dispositivo de entrada diferente ou uma nova forma de interação (por exemplo, através de sons captados por um microfone) pode tornar a experiência de jogar mais real ou divertida, ou ainda tornar o jogo acessível a usuários com necessidades físicas especiais.

Mouse, teclado, joystick e gamepad são os controladores mais comuns em jogos. Muitos jogos para computador dão suporte a esses controladores, mas não estão preparados para outros, como luvas de realidade virtual ou sistemas de rastreamento (*trackers*). Em alguns casos é possível tratar esses dispositivos como mouses ou joysticks sofisticados, mas às vezes essa simplificação impede que suas particularidades sejam bem exploradas.

Este trabalho propõe uma abordagem para desenvolver jogos que: (i) podem ser usados através de qualquer controlador e (ii) podem ser personalizados pelo jogador para permitir novas maneiras de jogar. A personalização de controles nos jogos atuais é muito limitada: o jogador pode configurar botões ou trocar um controlador por outro, mas, na maioria das vezes, fica restrito às possibilidades de interação de teclado, mouse e joystick.

O restante do artigo está organizado como se segue. A seção 2 cita trabalhos relacionados a controle. A seção 3 descreve alguns dispositivos de entrada ainda pouco usados em jogos. A seção 4 cita jogos que fazem uso de inovações em interatividade. A seção 5 expõe a nossa abordagem de desenvolvimento, enquanto a seção 6 mostra sua aplicação em alguns tipos comuns de jogos e a seção 7 apresenta as considerações finais deste trabalho.

2 Trabalhos relacionados

O DirectInput, o GlovePIE e o VRPN são bibliotecas que abstraem para a aplicação o tipo específico de controlador usado pelo usuário. Aliado à nossa abordagem, o uso dessas bibliotecas facilita a interface com uma grande quantidade de dispositivos de entrada.

DirectInput O DirectInput[1] é o módulo da biblioteca DirectX responsável por gerenciar a interface com os controladores. Ele é muito usado em jogos para o sistema operacional Windows e para os consoles Xbox e Xbox 360, da Microsoft. Por isso, muitos dispositivos voltados para usuários domésticos possuem drivers para DirectInput.

GlovePIE O GlovePIE, ou Glove Programmable Input Emulator [Kenner 2007], é um programa para Windows que emula teclado, mouse e joystick a partir de comandos de voz, gestos de mouse, luvas e *trackers*, entre outras possibilidades. Com ele, jogos que usam DirectInput podem ser controlados por dispositivos para os quais não foram programados.

VRPN O VRPN, ou Virtual Reality Peripheral Network [Russell M. Taylor et al. 2001], tem uma longa lista de controladores suportados, dos mais simples joysticks até dispositivos de realidade virtual como luvas e sistemas de rastreamento. Ele funciona como um servidor que recebe eventos de controladores e transmite-os para uma aplicação cliente via rede.

3 Controladores

Os controladores de jogo mais populares — teclado, mouse, joystick e gamepad — não exigem muito do jogador em termos de esforço físico: para usá-los, basta mexer os dedos ou, no máximo, as mãos. A forma de interação com esses controladores pode não refletir muito bem o que acontece dentro do jogo. Eis alguns outros dispositivos que podem ser usados em jogos:

Microfone É muito barato. Associado a algoritmos de reconhecimento de fala, pode ser usado para ativar comandos em jogos. Não exige coordenação motora e deixa as mãos livres para outras tarefas.

Webcam A utilização de técnicas de visão computacional para reconhecimento de padrões em imagens possibilita o uso de *webcams* em muitos jogos. Um exemplo são jogos de dança em que o jogador precisa movimentar os braços e pernas para posições mostradas na tela.

Sistema de rastreamento Ele captura o movimento de uma pessoa no espaço. Pode ser composto de duas ou mais câmeras e alguns marcadores. As câmeras detectam a posição dos marcadores, que são colados à pessoa ou a objetos usados por ela (óculos, por exemplo). Esses objetos, denominados alvos, podem ter diversos marcadores fixos, e isso permite ao sistema inferir não apenas a posição, mas também a orientação do alvo em relação às câmeras.

Wii Remote Também conhecido como Wiimote, é o controlador padrão do console Wii, da Nintendo (mas pode ser conectado a computadores através de uma interface Bluetooth). Ele é manipulado com uma mão e transmite ao console movimentos feitos pelo jogador. Para isso, ele conta com um acelerômetro, sensor que mede a aceleração nas três dimensões espaciais [WiiLi.org 2007]. Quando está parado, o acelerômetro registra a aceleração da gravidade, e com isso é possível saber a orientação do controle em relação ao chão. Além desse sensor, o Wii Remote possui botões convencionais e pode controlar um cursor ao ser apontado para a tela (através de um sensor de infravermelho específico para essa função).

4 Uso de controladores em jogos

Em *Black & White* [Lionhead 2001], o jogador lança magias através de gestos de mouse[2]. O jogo ajudou a popularizar essa técnica de controle.

[1] Disponível em http://msdn.microsoft.com/directx/

[2] Movimentos que descrevem um determinado padrão geométrico.

Existem jogos especificamente projetados para uso com *webcams*, tanto em consoles quanto em PCs [Sony 2003; Microsoft 2006; Creative 2007]. Em um deles, o jogador se mexe para estourar bolhas que se movimentam na tela.

No jogo *3D Language Spain*[3], o jogador pode conversar com habitantes de uma vila virtual através do microfone. O jogo avalia a pronúncia em espanhol do jogador, ajudando-o a aprender a linguagem.

O *WiiSports* é um pacote de jogos para o Wii que inclui golfe, beisebol, tênis, boliche e boxe. No tênis, por exemplo, o Wii Remote age como uma raquete: o jogador movimenta o braço para rebater e torce o pulso para dar efeitos na bola. A forma como o Wii Remote é movimentado influencia sutilmente a forma como o jogo responde.

5 Abordagem

A fim de facilitar o desenvolvimento de jogos para os mais diversos controladores, nossa abordagem propõe:

- a adoção de um modelo geral de controlador (*controlador virtual*);
- a identificação das ações do jogo e a concepção de um *controlador ideal* (hipotético) para o jogo;
- o reuso de soluções recorrentes de *mapeamento* entre diferentes eventos de controle.

5.1 Controladores virtuais

Chamamos de *controlador virtual* a representação abstrata de algum dispositivo de entrada. No nível de implementação, os controladores virtuais são classes que obedecem a uma interface bem definida.

A maioria dos controladores de jogos possui *componentes* de dois tipos básicos: botões e eixos. Um mouse simples, como o da figura 1(a), tem dois botões e controla a posição de um ponteiro na tela, descrita em função dos eixos X e Y de um plano cartesiano.

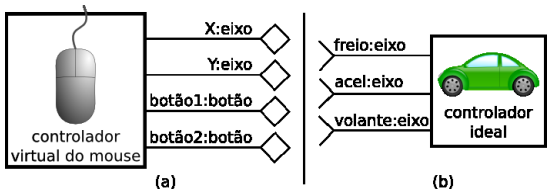


Figura 1: *Controlador virtual e controlador ideal*

Analogamente, um controlador virtual é um conjunto de botões e eixos. Um botão possui dois *estados*: pressionado ou não-pressionado. Já um eixo possui infinitos estados: ele pode assumir qualquer valor racional entre -1.0 e 1.0[4]. As mudanças de estado são chamadas de *eventos de controle*.

Os componentes podem ser acessados de duas formas: por amostragem ou por eventos. No primeiro caso, a aplicação obtém o estado do componente a qualquer momento. No segundo caso, a aplicação se inscreve em alguns controladores virtuais e recebe eventos deles à medida em que acontecem.

5.2 Controlador ideal de um jogo

Na nossa abordagem, assumimos que o controlador usado pelo jogador não é conhecido no tempo de desenvolvimento do jogo. Por isso, em vez de programar para controladores específicos, o desenvolvedor deve programar cada jogo para um *controlador ideal*. Trata-se de um controlador hipotético em que cada componente (botão ou eixo) está diretamente relacionado a uma ação do jogo.

Em um jogo de corrida, por exemplo, a ação de acelerar o carro poderia ser representada no controlador ideal por um botão, mas é mais adequado representá-la como um eixo — assim, é possível controlar a intensidade da aceleração. A depender do realismo desejado, o freio pode ser um eixo independente ou então ser acionado quando o eixo de aceleração assume valores negativos.

A figura 1(b) mostra a representação gráfica do controlador ideal de um jogo de corrida. Ele usa três eixos para controlar o carro: o freio, a aceleração e o volante.

5.3 Mapeamento

Controladores ideais são apenas abstrações. Do ponto de vista prático, jogadores manipulam dispositivos como o teclado e o mouse, e seus eventos são de alguma forma mapeados em eventos do controlador ideal. Olhando a figura 1, percebe-se que não é possível relacionar diretamente os componentes do mouse aos componentes do controlador ideal do jogo de carro: é preciso fazer um mapeamento mais elaborado.

Um *mapeador* é um módulo que recebe eventos de zero ou mais controladores virtuais e gera novos eventos de controle. Mapeadores podem ser combinados arbitrariamente para descrever mapeamentos complexos.

Com o mapeamento desacoplado da lógica do jogo, o projeto do jogo torna-se mais modular, trazendo benefícios ao seu desenvolvimento, como facilidade de manutenção, melhor compreensão do código-fonte e desenvolvimento paralelo de módulos. Em particular, favorece o reuso de soluções recorrentes para problemas de mapeamento, tornando mais fácil adaptar um jogo para usar controladores de maneiras não previstas.

Um jogo desenvolvido com nossa abordagem deve ter mapeamentos embutidos para os dispositivos mais comuns, mas, de preferência, também deve permitir que o jogador crie seu próprio mapeamento a partir de mapeadores simples usando uma interface gráfica.

Abaixo serão descritos alguns mapeadores comumente usados em diversos jogos, ainda que implicitamente. A descrição de cada mapeador segue um modelo bem definido, com os seguintes itens: (i) *nome*; (ii) *propósito*: descrição informal de sua funcionalidade; (iii) *parâmetros*: variáveis usadas na inicialização do mapeador; (iv) *entrada*: componentes esperados pelo mapeador; (v) *saída*: componentes gerados pelo mapeador; (vi) *comportamento*: descrição completa de sua funcionalidade; (vii) *exemplo de uso*: como pode ser usado em jogos.

1. **Nome**: Incremento
   Propósito: Tratar um eixo como valor relativo, usado para incrementar outro eixo.
   Parâmetros: $ciclico$:booleano, $fator$:real
   Entrada: a:eixo
   Saída: s_1:eixo
   Comportamento: s_1 é inicialmente zero. Então, a cada atualização, $s_1 = s_1 + fator \times a$. Quando o valor de s_1 ultrapassa as extremidades do intervalo $[-1.0, 1.0]$, o parâmetro $ciclico$ indica o que acontece: quando falso, s_1 assume o valor da extremidade ultrapassada; quando verdadeiro, assume o valor da extremidade oposta.
   Exemplo de uso: (i) Um ponteiro controlado por um joystick. A posição do joystick normalmente serve de incremento para a posição do ponteiro. Quando o ponteiro atinge os limites da tela, ele pára (não-cíclico). (ii) Em jogos de ação em primeira pessoa, o jogador pode girar para o lado indefinidamente, dando voltas em torno de si mesmo (cíclico).

2. **Nome**: Gatilho
   Propósito: Acionar um botão quando um eixo ultrapassa determinado valor.
   Parâmetros: $limiar$:eixo
   Entrada: a:eixo
   Saída: s_1:botão

[3] Disponível em http://www.garagegames.com/news/12877.

[4] Alguns dispositivos, como o mouse, adotam outras faixas de valores. Assumimos que os valores são convertidos para a faixa -1.0 a 1.0 antes de serem usados pela aplicação.

Comportamento: Quando $a > limiar$, o botão s_1 é pressionado e só é liberado quando, depois disso, $a <= limiar$.
Exemplo de uso: Jogo de golfe do *WiiSports*. A tacada só é dada quando a aceleração do Wii Remote ultrapassa um certo valor mínimo.

3. **Nome**: Botões»Eixo
Propósito: Controlar um eixo usando dois botões.
Parâmetros: (nenhum)
Entrada: a:botão, b:botão
Saída: s_1:eixo
Comportamento: Quando somente a está pressionado, $s_1 = -1.0$. Quando somente b está pressionado, $s_1 = 1.0$. Caso contrário, $s_1 = 0.0$
Exemplo de uso: Jogo de corrida controlado pelo teclado. Usam-se as setas direcionais pra controlar o volante.

4. **Nome**: Oscilador
Propósito: Gerar valores entre -1.0 e 1.0, variando no tempo.
Parâmetros: v:real
Entrada: (nenhuma)
Saída: s_1:eixo
Comportamento: A saída s_1 oscila entre -1.0 e 1.0 com velocidade v.
Exemplo de uso: Jogo de golfe controlado por teclado ou mouse. Nesse caso, é comum ter uma barra de força oscilando rapidamente. Quando o jogador pressiona algum botão, o valor da barra de força é usado para controlar a força da tacada.

5. **Nome**: Combo
Propósito: Acionar um botão quando uma seqüência de botões é pressionada.
Parâmetros: $vetB$:botão[]
Entrada: a:botão
Saída: s_1:botão
Comportamento: Detecta quando a seqüência de botões $vetB$ é pressionada e, então, aciona o botão s_1.
Exemplo de uso: Combo de botões para golpes especiais em jogos de luta.

6. **Nome**: Gesto
Propósito: Reconhecer um desenho em duas dimensões.
Parâmetros: $vetA$:eixo[],$vetB$:eixo[]
Entrada: a:eixo, b:eixo
Saída: s_1:botão
Comportamento: Os eixos a e b são considerados coordenadas (X, Y) na tela. Quando essas coordenadas formam, aproximadamente, o desenho descrito pelos parâmetros $vetA$ e $vetB$, o botão s_1 é acionado.
Exemplo de uso: Atalhos para lançar magias no jogo Black & White.

Considerando que o estado de um botão é um valor binário, pode-se definir mapeadores para os operadores booleano E, Ou e Não. O mapeador E, por exemplo, recebe o estado de dois ou mais botões e gera, como saída, o estado "pressionado" apenas quando todos os botões da entrada estão pressionados. Analogamente, é possível definir mapeadores como Soma, Produto, Média, Máximo e Mínimo, que recebem dois ou mais eixos e retornam um eixo.

6 Estudos de caso

Para exemplificar nossa abordagem, criamos mapeamentos para dois tipos de jogos: tiro em primeira pessoa (FPS, do inglês *first-person shooter*) e golfe. Levamos em consideração diferentes conjuntos de controladores.

6.1 FPS

Em um jogo típico no estilo FPS, o jogador pode deslocar-se para os lados, para a frente e para trás, e também girar a câmera para os lados, para cima e para baixo. A mira fica sempre no centro da tela, como se girasse junto com a câmera. Além disso, o jogador pode pular e atirar.

O controlador ideal esboçado na figura 2 apresenta um botão para pular, um botão para atirar, dois eixos para deslocamento (`velX` e `velY`), dois eixos que indicam a orientação da câmera, em termos absolutos (`rotZ` e `rotX`) e também dois eixos para controlar a posição da mira em relação ao centro da tela (`miraX` e `miraY`).

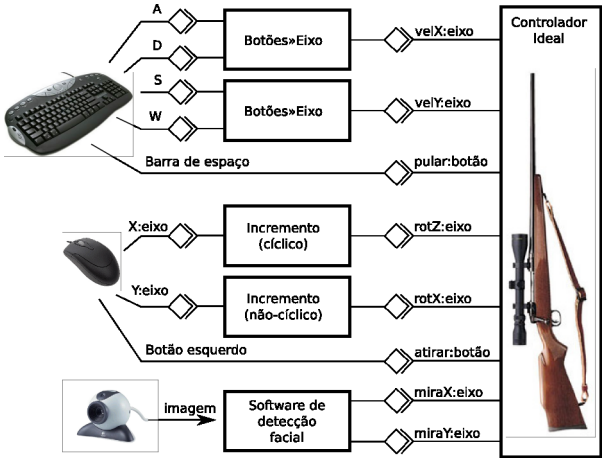


Figura 2: *Mapeamento para jogo FPS*

Os botões A e D movimentam o jogador para os lados, e os botões W e S, para frente e para trás. O mapeador `Botões»Eixo` é responsável por transformar cada par de botões em um eixo, que assume os valores -1.0, 0.0 ou 1.0. O deslocamento do mouse na horizontal (eixo X) faz a câmera girar para os lados. Esse movimento é cíclico, isto é, a câmera pode girar indefinidamente, dando voltas em torno de si mesma. Já o deslocamento na vertical (eixo Y) faz a câmera girar para cima e para baixo, dentro de certos limites (é um movimento, portanto, não-cíclico).

A novidade fica por conta da mira, que, no nosso controlador ideal, pode se mover na tela. No esquema da figura, uma *webcam* é usada para filmar o jogador. Um *software* detecta a posição do rosto do jogador e a descreve em coordenadas X e Y, que são usadas para posicionar a mira no jogo.

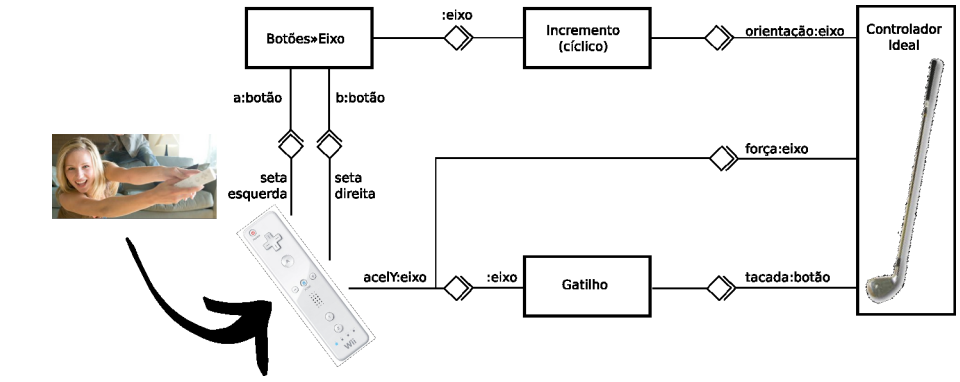
6.2 Golfe

Imaginamos um jogo de golfe simples, em que o jogador pode girar a câmera para os lados e dar uma tacada com determinada força. Para simplificar o exemplo, consideramos que apenas um modelo de taco está disponível. O controlador ideal é composto de um eixo para a `orientação`, outro para a `força` da tacada e um botão para dar a `tacada`. A `orientação` é interpretada pelo jogo como um ângulo absoluto entre 0° e 360°. Uma barra de força, mostrada na tela, é constantemente atualizada com o valor do componente `força` do controlador, e pára quando o botão `tacada` é pressionado.

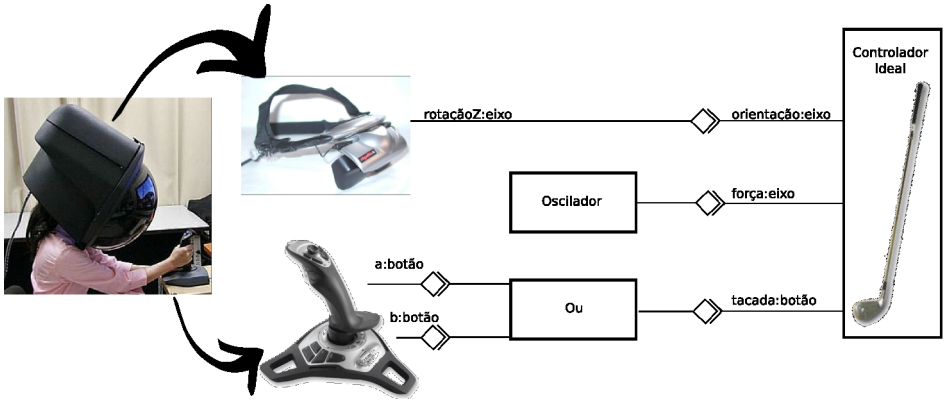
Note que a ação de dar a tacada foi separada nos componentes `força` e `tacada`. Com isso é possível aproveitar as particularidades de diversos dispositivos, como mostram as figuras 3(a) e 3(b).

Na figura 3(a), o jogador usa apenas um Wii Remote. As setas direcionais giram o jogador para os lados. Primeiramente, as setas são mapeadas para um eixo, que é então usado para incrementar ou decrementar a orientação do jogador. A aceleração do Wii Remote controla a barra de força (no exemplo, consideramos a aceleração em um eixo). Quando essa aceleração ultrapassa determinado limiar, o botão `tacada` é disparado.

Em outra situação, ilustrada na figura 3(b), o jogador usa óculos de realidade virtual e pode girar a cabeça livremente para os lados. Um sistema de rastreamento mede a rotação dos óculos ao redor do eixo Z, perpendicular ao chão, e essa informação é transmitida ao jogo como a orientação do jogador. A barra de força oscila sozinha (graças ao controlador virtual `Oscilador`). Cabe ao jogador pressionar ao menos um dos dois botões do joystick para dar a tacada no momento em que o valor da barra de força for adequado.



(a) Usando o Wii Remote



(b) Usando um tracker e um joystick

Figura 3: *Mapeamentos para o jogo de golfe*

7 Considerações finais

Este trabalho apresentou uma nova abordagem para a definição de controles de um jogo, que estimula o desenvolvimento da interface de controle dissociado de particularidades de controladores específicos. Através do reuso de mapeadores comuns, a abordagem torna possível personalizar um jogo para ser usado com diversos dispositivos de entrada, aproveitando as possibilidades de interação inerentes a cada um deles.

É claro que a jogabilidade depende dos controladores utilizados. Controlar apenas com o teclado a mira da arma de um jogo FPS, por exemplo, seria difícil e desagradável. Por outro lado, restringir as ações de um FPS ao que pode ser controlado com um teclado, um mouse e duas mãos limita a jogabilidade quando o usuário dispõe de controladores mais sofisticados, como um Wii Remote ou um sistema de rastreamento. Com nossa abordagem, a interação com um jogo através de controladores inadequados é ao menos possível (embora talvez entediante); já a interação através de controladores avançados pode explorar todo o potencial do jogo e tornar a experiência de jogo bastante rica e realista.

Em um cenário ideal, os mapeamentos poderão ser personalizados pelo próprio jogador, que, através de uma interface gráfica, esboçará esquemas como aqueles apresentados neste artigo. Outra possibilidade é permitir o uso de *scripts*, destinados a usuários avançados, para configurar os mapeamentos com toda a expressividade de uma linguagem de programação. Em ambos os casos, os mapeamentos poderão ser salvos e compartilhados com outros jogadores.

Planejamos implementar essa abordagem no motor InGE [Bessa et al. 2007] e, então, criar diversos mapeamentos para o jogo TampCross[5]. A partir daí, podemos fazer experiências para avaliar a usabilidade de cada controle no jogo.

[5]Disponível em http://indigente.dcc.ufba.br/?q=tampcross.

Agradecimentos

A Christina, pelas sugestões para melhorar o artigo.

Referências

BESSA, A., SOUSA, C. T., BEZERRA, C. E., MONTEIRO, I., BANDEIRA, H., AND SOUZA, R. 2007. Desenvolvendo um Motor Multiplataforma para Jogos 3D. In *SBGames 2007*.

CREATIVE, 2007. WebCam Game Star. Disponível em http://www.creative.com/. Último acesso em 23 de agosto de 2007.

KENNER, C., 2007. GlovePIE. Disponível em http://carl.kenner.googlepages.com/glovepie.

LIONHEAD, 2001. Black & white. Último acesso em 17 de Agosto.

MICROSOFT, 2006. Xbox LIVE Vision Camera. Disponível em http://www.xbox.com/en-US/hardware/x/xboxlivevision/. Último acesso em 23 de agosto de 2007.

MYERS, B., HUDSON, S. E., AND PAUSCH, R. 2000. Past, present, and future of user interface software tools. *ACM Trans. Comput.-Hum. Interact. 7*, 1, 3–28.

RUSSELL M. TAYLOR, I., HUDSON, T. C., SEEGER, A., WEBER, H., JULIANO, J., AND HELSER, A. T. 2001. VRPN: a device-independent, network-transparent VR peripheral system. In *VRST '01: Proceedings of the ACM symposium on Virtual reality software and technology*, ACM Press, New York, NY, USA, 55–61.

SONY, 2003. EyeToy. Disponível em http://www.eyetoy.com/. Último acesso em 23 de agosto de 2007.

WIILI.ORG, 2007. Wiimote. Disponível em http://www.wiili.org/index.php?title=Wiimote&oldid=9285. Último acesso em 22 de agosto de 2007.